PASSED

Computer lab 6, Group 4

Kevin Neville, Gustav Sternelöv, Vuong Tran

14 mars 2016

Assignment 1: Genetic algorithm

In this assignment, you will try to perform one-dimensional maximization with the help of a genetic algorithm.

1.1. Define the function $f(x) = \frac{x^2}{e^x} - 2e^{\frac{-9\sin(x)}{x^2+x+1}}$.

```
# 1.1
func <- function(x) {
  res <- ((x^2) / exp(x)) - 2 * exp((-9*sin(x)) / (x^2 + x + 1) )
  return(res)
}
```

1.2. Define the function “crossover” that for two scalars x and y returns their “kid” as (x+y)/2

```
# 1.2
f_crossover <- function(x, y) {
  res <- (x + y) / 2
  return(res)
}
```

1.3. Define the function “mutate” that for scalar x returns the result of the integer division x2 mod 30. (Operation “mod” is denoted in R as “%%”)

```
# 1.3
f_mutate <- function(x) {
  res <- (x^2) %% 30
  return(res)
}
```

1.4. Write a function that depends on the parameters maxiter and mutprob and:

a. Plots function f in the range from 0 to 30. Do you see any maximum value?

b. Defines an initial population for the genetic algorithm as X=(0,5,10,15,...30)

c. Computes vector “Values” that contains the function values for each population point

d. Performs maxiter iterations where at each iteration

i. Two indexes are randomly sampled from the current population, they are further used as parents (use sample().)

ii. One index with the smallest objective function is selected from the current population, the point is referred to as victim(use order())

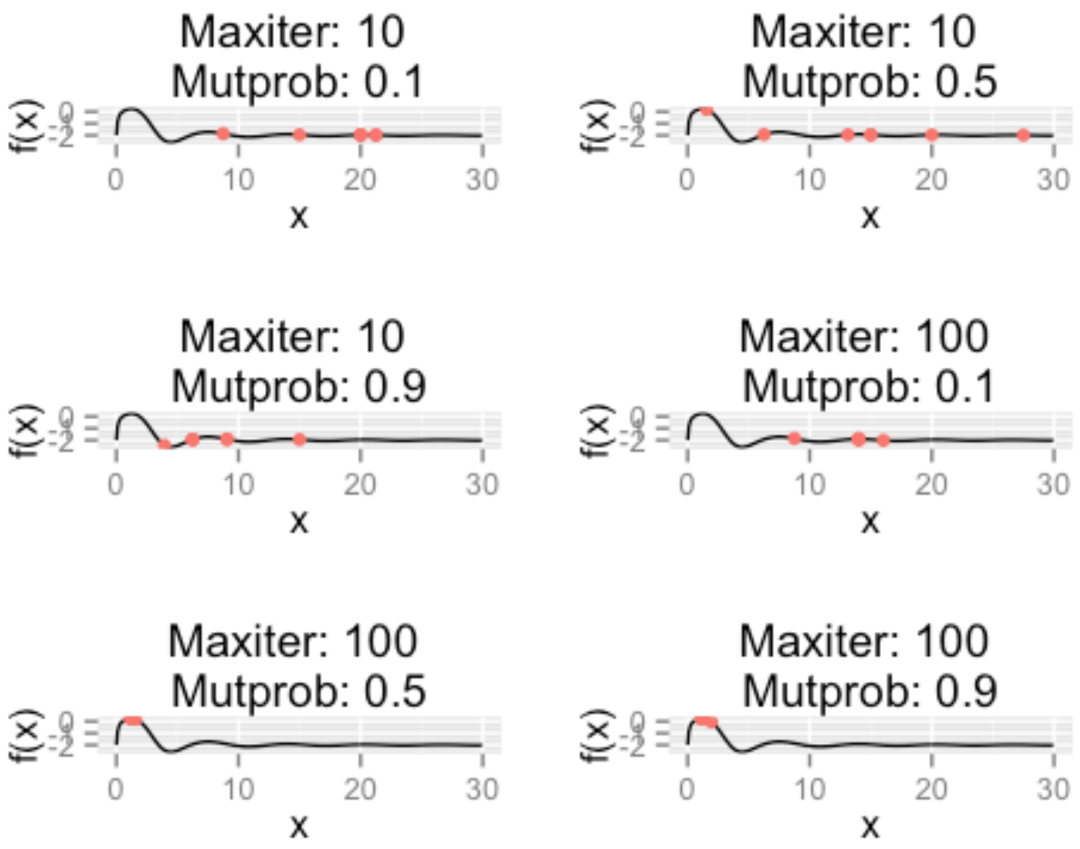
iii. Parents are used to produce a new kid by crossover. Mutate this kid with probability mutprob. (use crossover(), mutate())

iv. The victim is replaced by the kid in the population and the list “Values” is updated

v. The current maximal value of the objective function is saved.

e. Final observations are added to the current plot and marked by some other color. 5. Run your code with different combinations of maxiter=10, 100 and mutprob=0.1, 0.5,0.9.

Observe the initial population and final population. Conclusions?



The max value of the initial objective function is 0.20. This can be seen in the graphs above.

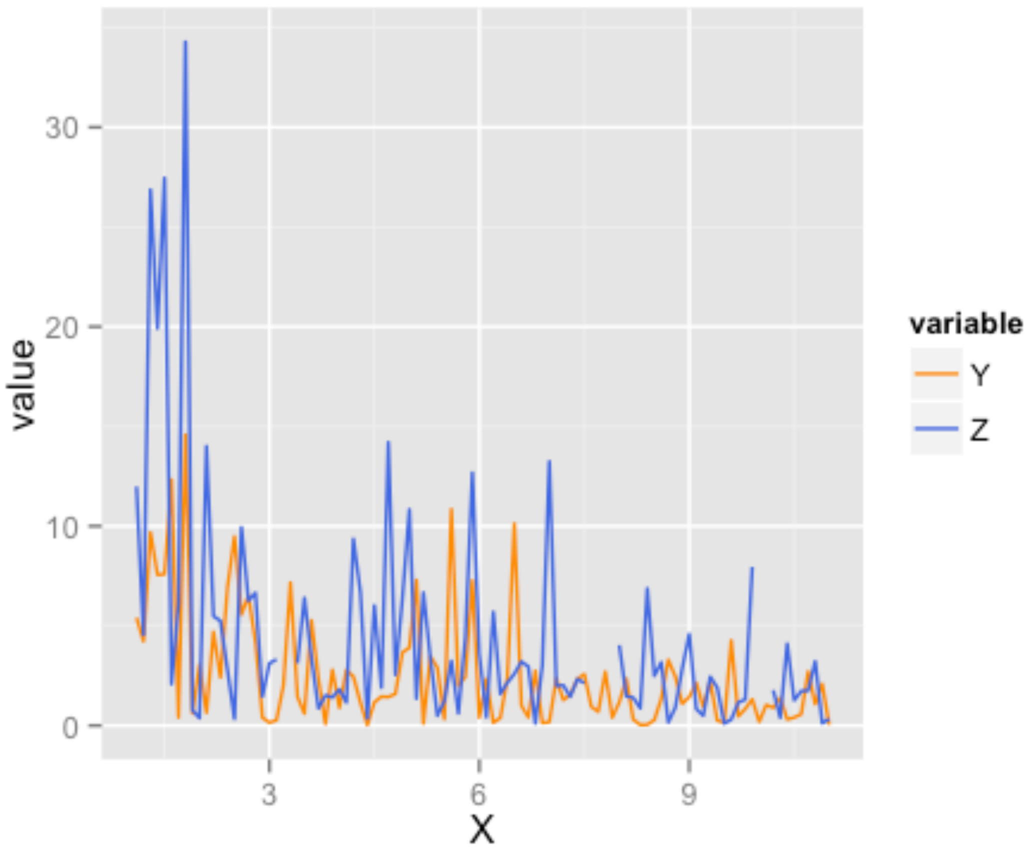
When using maxiter=10 the final population is similar to the initial population. This pattern can be noticed for the three different values of mutprob. However when using mutprob=0.5 we get one point which is close to the maximum value for the objective function, this is problaby more due to a bit of luck rather than an effect of the mutprob value.

The initial population and the final population differ alot when using 100 iterations. The majority of the points are very close to the maximum values. When using a higher mutprob the function allows the population to change more often. There is a trend that shows that higher mutation probabilty leads to more accurate estimates of the maximum value.

Assignment 2: EM algorithm

2.1 Data file physical.csv describes a behavior of two related physical processes Y = Y(X) and Z = Z(X).

2.1 Make a time series plot describing dependence of Z and Y versus X. Does it seem that two processes are related to each other? What can you say about the variation of the response values with respect to X?



The two process seem to follow eachother well. The Y process seem to have less varaition compared to the Z process. This because the Y process has values that differ more extremely between high and low values compared to the Z process.

2.2. Note that there are some missing values of Z in the data which implies problems in estimating models by maximum likelihood. Use the following model:

$$\mathrm{Y_i} \sim \mathrm{Exp}(\frac{\mathrm{X_i}}{\lambda}), \mathrm{Z_i} = \mathrm{Exp}(\frac{\mathrm{X_i}}{2\lambda})$$

where λ is some unknown parameter to derive an EM algorithm that estimates λ.

Given models:

$Y_i \sim Exp\left(\frac{x_i}{\lambda}\right)$,
$Z_i \sim Exp\left(\frac{x_i}{2\lambda}\right)$

We search for the marginal distribution:
$P(Y,Z|\lambda) = \prod_{i=1}^{n} \frac{x_i}{\lambda} * \exp\left(-\frac{1}{\lambda}(\,Y_i * X_i)\right) * \prod_{i=1}^{n} \frac{x_i}{2\lambda} * \exp\left(-\frac{1}{2\lambda}(\,Z_i * X_i)\right)$

$$P(Y,Z|\lambda) = \prod_{i=1}^{n} \frac{x_i^2}{2\lambda^2} * \exp\left(-\frac{1}{2\lambda} X_i(2\,Y_i + Z_i)\right) = \left(\frac{1}{2\lambda^2}\right)^n * \prod_{i=1}^{n} X_i^2 * exp(-\frac{1}{2\lambda}\sum_{i=1}^{n}(X_i(2Y_i + Z_i))$$

$$l(p) = \log\,(\,P(Y,Z|\lambda) = nlog\left(\frac{1}{2\lambda^2}\right) + \sum_{i=1}^{n} \log(X_i^2) - \frac{1}{2\lambda}\sum_{i=1}^{n} X_i(2Y_i + Z_i)$$

Since Z has missing values, we cannot use maximum likelihood estimation, instead we will try to use Expectation maximum (EM) algorithm.

We start off with splitting up Z in two subsets:

$Z_o = \{Z_i; i \;\in Observed\}$

$Z_M = \{Z_i; i \;\in Missing\}$

$$E\big(l(p)\big) = \; nlog\left(\frac{1}{2\lambda^2}\right) + \sum_{i=1}^{n} \log(X_i^2) - \frac{1}{2\lambda}\sum_{i=1}^{n} X_i\big(2Y_i + (Z_o + Z_M)\big) =$$

$$nlog\left(\frac{1}{2\lambda^2}\right) + \sum_{i=1}^{n} \log(X_i^2) - \frac{1}{\lambda}\sum_{i=1}^{n} X_i * Y_i - \frac{1}{2\lambda}\sum_{i\in O} X_i * Z_o - \frac{1}{2\lambda}\sum_{i\in M} X_i * Z_M$$

$Since\ Z_M\ is\ missing\ values\ we\ expect\ it\ to\ be\ mean\ values\ from \sim Exp\left(\frac{X_i}{2\lambda}\right) = (\frac{2\lambda}{X_i})$

So, $\sum_{i \in M} X_i * Z_M = \sum_{i \in M} X_i * \left(\frac{2\lambda_t}{X_i}\right) = \sum_{i \in M} 2\lambda_t = |M| * 2\lambda_t$

$$\frac{\partial E\big(l(p)\big)}{\partial \lambda} = 0:$$

$$-\frac{2n}{\lambda} + \frac{\sum_{i=1}^{n} X_i * Y_i}{\lambda^2} + \frac{\sum_{i \in O} X_i * Z_o}{2\lambda^2} + \frac{\lambda_t * |M|}{\lambda^2} = 0 \to \cdots \to$$

$$\to \hat{\lambda} = \lambda_{t+1} = \frac{2\sum_{i=1}^{n} X_i * Y_i + \sum_{i \in O} X_i * Z_o + 2(\lambda_t * |M|)}{4n}$$

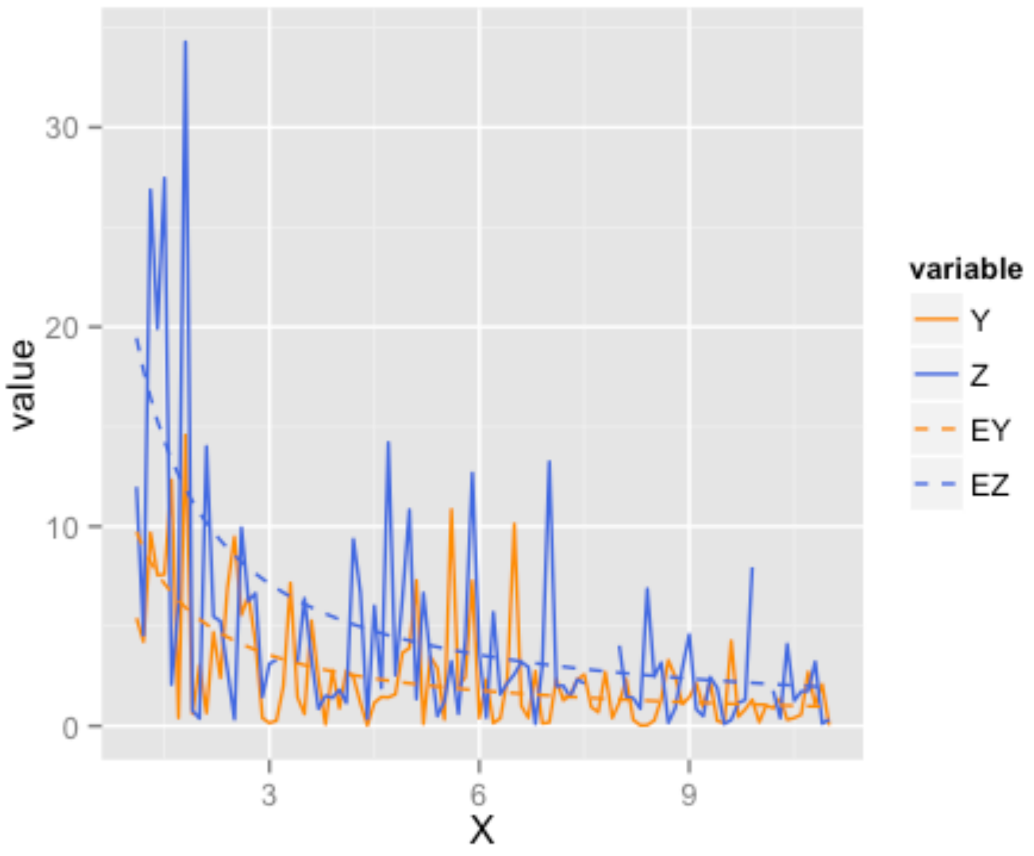
Where |M| is amounts of missing values.

2.3 Implement this algorithm in R ad use λ_0=100 and convergence criterion "stop if the change is λ is less than 0.001". What is the optimal λ and how many iterations were required to compute it?

```
## 100 1
## 14.26782 2
## 10.83853 3
## 10.70136 4
## 10.69587 5
```

As can be seen in the table above we obtain the optimal λ=10.695875, this occurs after five iterations.

2.4 Plot EY and EZ versus X in the same plat as Y and Z versus X and comment wheter the computed λ seems to be reasonable.



The computed expected values seems to be reasonable estimates of the expected value. Which means that our estimated values of lambda are good.

Contributions

All of the group members have contributed to this lab report. The code is a mixture of all the group members own code. The majority of the text is compiled together.

Appendix - R-code

```
library(ggplot2)
# 1.1
func <- function(x) {
```

```r
  res <- ((x^2) / exp(x)) - 2 * exp((-9*sin(x)) / (x^2 + x + 1) )
  return(res)
}
# 1.2
f_crossover <- function(x, y) {
  res <- (x + y) / 2
  return(res)
}
# 1.3
f_mutate <- function(x) {
  res <- (x^2) %% 30
  return(res)
}
# 1.4
gen.func <- function(maxiter, mutprob) {
  # a
  gg_vals <- data.frame(vals=func(seq(from=0, to=30, by=0.1))) #ggplot
  gg_vals$x <- seq(from=0, to=30, by=0.1)

  # b
  init_pop <- seq(from=0, to=30, by=5)

  # c
   vec_val <- func(init_pop)

  # d
   max_v <- c()
  for(i in 1:maxiter) {
  # i
  parents <- sample(init_pop, 2, replace = FALSE)

  # ii
  index <- order(vec_val)[1]
  victim <- init_pop[index]

   # iii
   kid <- f_crossover(parents[1], parents[2])

   # iv
   if(runif(1,0,1) < mutprob){
     init_pop[index] <- f_mutate(kid)
   } else {
     init_pop[index] <- kid
   }
  vec_val[index] <- func(init_pop[index])
   # v

   max_v[i] <- max(func(init_pop))
  } # end of for loop
  # e

  gg_init_pop <- data.frame(y=func(init_pop), x=init_pop)

 gen.plot <-  ggplot(gg_vals, aes(x=x, y=vals)) + geom_line() +
    geom_point(data = gg_init_pop, aes(y=y, x=x, colour="red")) +
   labs(title=paste("Maxiter:", maxiter, "\nMutprob:", mutprob, sep = " "),
        y="f(x)") + theme(legend.position="none")

  return(list(current_pop = init_pop, f_x=func(init_pop), max_v=max_v, gen.pl
ot=gen.plot))
} # end of function
library(gridExtra)
set.seed(12345)
p1 <- gen.func(10, 0.1)$gen.plot
set.seed(12345)
p2 <- gen.func(10, 0.5)$gen.plot
set.seed(12345)
p3 <- gen.func(10, 0.9)$gen.plot
set.seed(12345)
p4 <- gen.func(100, 0.1)$gen.plot
set.seed(12345)
p5 <- gen.func(100, 0.5)$gen.plot
set.seed(12345)
p6 <- gen.func(100, 0.9)$gen.plot
grid.arrange(p1, p2, p3,
             p4, p5, p6, ncol=2)
physical <- read.csv("/Users/Kevin/Desktop/Computational-statistics/Lab6/phys
ical.csv")

library(reshape2)
gg_physical <- melt(physical, id="X")
ggplot(data = gg_physical, aes(x=X, y=value, colour= variable)) + geom_line()
+
scale_colour_discrete(name  ="Physical processes") +
  scale_color_manual(values = c("darkorange", "royalblue"))
em.exp <- function(X,Y,Z){
  Zobs <- Z[!is.na(Z)]
  Zmiss <- Z[is.na(Z)]
  n <- length(c(Zobs, Zmiss))
  r <- length(Zobs)
  Xobs<- X[!is.na(Z)]
  Xmiss <- X[is.na(Z)]
  # Initial values
lambda=100

i=0
  repeat{
    # E-step
      #don't need here
    # M-step
    #lambda1
    lambda1<-(2*sum(X*Y)+sum(Zobs*Xobs)+2*(lambda*(n-r)))/(4*n)
    # Update parameter values
    #lambda=lambda1
    i=i+1

    # Compute log-likelihood using current estimates
    #llt <- ll(z=Zobs, lambda = lambda, n=n,x=Xobs)
    # Print current parameter values and likelihood
    cat(lambda, i, "\n")
    # Stop if converged
    if ( abs(lambda1 - lambda) < 0.001) break
    lambda=lambda1
  }
return(lambda)
}
bestLambda<-em.exp(physical$X, physical$Y, physical$Z)
#2.4
physical$EY <- bestLambda/physical$X
physical$EZ <- 2* bestLambda/physical$X
gg_physical <- melt(physical, id="X")

ggplot(gg_physical, aes(x=X, y=value, colour=variable, linetype=variable)) +
geom_line() +
  scale_color_manual(values = c("darkorange", "royalblue", "darkorange", "roy
alblue")) +
  scale_linetype_manual(values = c("solid", "solid", "dashed", "dashed"))
## NA
```